Administración de Sistemas Informáticos

Curso 2017-2018

Enzo Ferey - enzo.ferey@alumnos.upm.es

Introducción

Sistema operativo: pieza de software fundamental de la máquina, “intermediario” entre el hardware y las aplicaciones. Proporciona abstracción del hardware, gestiona memoria, procesos, dispositivos de entrada/salida... y arbitra el acceso a recursos.

UNIX: familia de sistemas operativos que derivan del UNIX original. Todas ellas siguen una filosofía común y pueden pasar o no los estándares SUS (Single UNIX Specification). Así harán parte de esta familia:

- UNIX originales: la mayoría de los UNIX comerciales.
- UNIX de marca: cumplen estándar SUS pero han sido modificados. Max OS X.
- UNIX de factor: no certificados pero que comparten la filosofía. GNU/Linux.

UNIX

Fundamentos

El pilar fundamental de todo sistema operativo UNIX es su línea de comandos (shell). En cuanto a la administración de sistemas nos concentramos en el scripting, programas sencillos ejecutados por el propio shell. Permiten automatizar tareas repetitivas y controlar funciones cotidianas del S.O. de manera homogénea.

Servicios del sistema: aplicaciones que están ejecutándose de manera permanente en el sistema. Proporcionan funciones de nivel a usuarios y aplicaciones y muchos de ellos se controlan mediante scripts.

En UNIX, todos los dispositivos de almacenamiento de archivos se muestran en un único árbol de directorios que contienen los siguientes directorios:

- **/bin**: ejecutables básicos del S.O.

- **/dev**: ficheros especiales asociados a dispositivos. Se proyectan los dispositivos como ficheros en este directorio para tener una interfaz común. Todos ellos tienen

asociado un major y minor number para determinar cómo tratar el acceso al dispositivo. El major identifica el driver (/proc/devices) y el minor es un parámetro adicional.

- **/etc**: configuración del sistema. Servicios, arranque, etc.
- **/home**: cuentas personales de los usuarios generalmente divididos por niveles.
- **/lib**: bibliotecas básicas del sistema. Propias del kernel o comunes a muchos ejecutables.
- **/mnt**: directorio de montaje de ciertos sistemas de ficheros (otros dispositivos).
- **/opt**: aplicaciones adicionales del sistema. Todo aquello que el administrador instale fuera del SSOO.
- **/proc**: sistema de ficheros virtual para la gestión de recursos. Procesos en ejecución, información del sistema, mapping de recursos del sistema. Al registrar una entrada se guarda una pareja de funciones de lectura y escritura para que las posteriores peticiones sean realizadas dentro del kernel y no al dispositivo. Además contiene un directorio con el PID de cada proceso en ejecución.
- **/sbin**: ejecutables de administración del SSOO. Subconjuntos de mandatos con privilegio. En un principio un usuario no tiene por qué tenerlos en el PATH.
- **/tmp**: directorio para ficheros temporales.
- **/usr**: aplicaciones adicionales del SSOO. Contienen los mandatos que no son básicos (ls, cp, etc.) pero que se distribuyen con la instalación del fabricante (sino irían en el directorio /opt).
- **/usr/local**: análogo a /opt, de hecho a veces es un enlace simbólico.
- **/var**: directorio para ficheros de log y colas de trabajo.

Usuarios, grupos y permisos

Cada usuario del sistema está identificado por un UID único, al que ve asociado el nombre de usuario (palabra en minúsculas) también único.

Todo usuario pertenece al menos a un grupo y la pareja de valores sirven para determinar los permisos de acceso a los ficheros (r, w, x).

Recordemos que que estos permisos son también aplicables a directorio y sirven para listar el contenido, crear nuevos archivos y entrar al directorio respectivamente.

Todos los archivos tienen tres categorías de permisos: al propietario, al grupo y “al mundo”. Algunos mandatos relacionados:

- **chown**

  Permite cambiar el propietario de uno o más ficheros.
  E.g: chown usuario2 archivo1

- **chgrp**

  Similar a chown pero para grupo.
  E.g: chown grupo2 archivo1

- **chmod**

  Permite cambiar los permisos de uno o más ficheros.
  E.g:

chmod 755 archivo1	user => r (4) + w (2) + x (1) group => r(4) + x(1) others => r(4) + x(1)
chmod uo+w archivo1	add write permission for user and others
chmod ug+rx archivo1	add read and execute permission for user and group
chmod a-w archivo1	remove write permission for all

Pasos en creación de un usuario

1) Insertarlo en el fichero de usuario. Se le asigna nombre, identificador y directorio de trabajo (usuario:passwd:uid:gid:desc:home:shell).
2) Asignación de contraseña.
3) Definir límites de número máximo de procesos, memoria, etc.
4) Crear el directorio home.
5) Copiar fichero iniciales normalmente situados en /etc/skel a modo de esqueleto.
6) Darle de alta en otros servicios disponible como mail o quota.

Todos estos pasos se pueden hacer con el comando adduser. E.g.:

```
adduser --home /home/user1 --shell /bin/bash --ingroup users user1
```

Para cambiar de usuario existe el mandato su y existe el mandato sudo para ejecutar un comando bajo la identidad del usuario root.

Por último una utilidad que se utilizará mucho en las tareas de administración de sistema son los mandatos de ejecución automática y programada como son cron y crontab.

Respecto al primero se trata de un demonio que ejecuta periódicamente los mandatos que se encuentren en los directorios /etc/cron.daily, /etc/cron.weekly y /etc/cron.monthly (el nombre indica la periodicidad).

Respecto al segundo permite gestionar ficheros crontab para cada usuario y que así éste pueda controlar más específicamente una serie de script que ejecutar según el siguiente formato:

```
[ minuto 0-59 ] [ hora 0-23 ] [día mes 1-31 ] [mes 1-12] [ día semana 0-7 ] mandato
```

E.g.:

```
0 0 * * * backup.sh
5 5 * * 7 system check.sh
0 2 1 * * software update.sh
```

Programación en bash

Las variables se crean directamente como nombre=valor. Se distinguen minúsculas y mayúsculas. Si se prefija el mandato export se convierten en variables de entorno (el mandato env sirve para listar todas las variables de entorno). Para acceder a su valor se coloca el prefijo $.

Todas las variables se tratan como cadenas de texto. Esto es VAR=1+2 valdrá la cadena "1+2". Para evaluar matemáticamente es necesario anidar la operación de la siguiente manera VAR=$((1+2)), de esta forma la variable valdrá "3".

Algunas variables de entorno predefinidas son:

- PATH: rutas donde buscar ejecutables.
- PWD: directorio actual.
- UID: uid del usuario.
- ?: valor de retorno del último mandato invocado.
- RANDOM: generador de números aleatorios.

Bash permite redirigir la entrada y salida estándar de los procesos que crea. El operador < se usa para redirigir la entrada de un mandato a un fichero. El operador > se usa para redirigir la salida de un mandato a un fichero. Se puede especificar descriptor a redirigir e.g: mandato 2> fichero y también se puede especificar operación de append de la salida con mandato >>.

Se pueden expandir comandos con la tilde grave ``.
E.g:

```
mandato1 param1 `mandato2 param2 param3` param4
```

Se pueden usar comillas simples o comillas dobles para representar cadenas de texto, pero sólo las comillas dobles permiten expansión de variables tal que "${NOMBRE}".

En una secuencia de mandatos se puede unir cada uno de ellos con:

- Punto y coma (;): concatenación normal.
- AND (&&): un mandato se ejecuta si el anterior devolvió un 0.
- OR (||): un mandato se ejecuta si el anterior no devolvió un 0.
- Pipe (|): la salida estándar del proceso a la izquierda se redirige a la entrada estándar del de la derecha.

Expasión de nombre de fichero

Los caracteres `*`, `?`, `[`, `]` se usan para reemplazar un patrón por la lista ordenada de todos los ficheros en el directorio actual que encajan con el patrón:

Todos los ficheros de extensión `txt`

```
ls *.txt
```

Todos los ficheros con cualquier extensión de 3 caracteres:

```
ls *.???
```

Todos los ficheros que empiezan por `a`, `b` o `c`:

```
ls [abc]*
```

Expansión de llaves

Útil para escribir secuencias repetitivas de argumentos:

```
{a,b,c}d → ad bd cd
fichero{1..3} → fichero1 fichero2 fichero3
{a..b}{1..2} → a1 b1 a2 b2
```

Expansión con substitución

Expansión de variable con sustitución con la construcción `${var/patron/sub}`:

```
VAR=fichero.png
EXT=.jpeg
${VAR/.png/$EXT} → fichero.jpeg
```

if

Condición `if` típica

```
if <condición>
then
    mandato1
    ...
elif <condición>
then
    ...
else
    ...
fi
```

- Las condiciones son a su vez secuencias de mandatos, que se interpretan en función de su valor de retorno
- El sub-bloque `else` se puede omitir
- El sub-bloque `elif` puede aparecer 0 ó más veces

```
if <cond>; then mandato1; ...; else mandato2; ...; fi
```

Un mandato muy habitual es el uso de `test` que permite evaluar ciertas condiciones lógicas `test condicion`. También se puede sustituir por cochertes `[ condicion ]`.

case

```
case <expresión> in
    caso_1 )
        mandato1
        mandato2;;
    caso_2 )
        mandato3
        ...;;
    ...
esac
```

Se evalúa el resultado de la expresión, no su retorno

while

```
while <condición>
do
    ...
done
```

until

```
until <condición>
do
    ...
done
```

for

```
for variable in <lista>
do
    ...
done
```

- La lista está formada por una secuencia de una o más cadenas separadas por espacios

select

```
select variable in <lista>
do
    ...
    [break]
done
```

- La sentencia `select` muestra un pequeño menú con las opciones de la lista. `break` se emplea para romper el bucle

Mandatos útiles

- head [file]: muestra las primeras líneas de un fichero.
- tail [file]: muestras las últimas líneas de un fichero. Si no se indica fichero operan con la entrada estándar.
- seq [number]: genera una secuencia de números.
- wc [file]: cuenta líneas, palabras y bytes de un fichero.
- cut [options]: selecciona columnas específicas en ficheros de texto.
- grep [options] pattern [file …]: muestra las líneas de un fichero que presentan un patrón.
- sed [options]: realiza transformaciones sobre cadenas de texto.
- test [options]: realiza comprobaciones y evalúa condiciones lógicas. -f ~= fichero existe.

Un bash script es un fichero de texto plano que contiene una serie de mandatos y operadores que pueden ser interpretados y procesados por bash. Para ejecutarlos es necesario invocar a bash como un mandato tal que `bash my_script.sh`. También se puede incluir la cabecera `#!/bin/bash` al principio del fichero para poder ejecutarlo como un programa más `./my_script.sh`

Estructura básica

Un *script* de bash contiene líneas con declaraciones de variables, secuencias de mandatos, sentencias de control...
Ejemplo:

```
#!/bin/bash
# Leemos los parámetros de entrada
CADENA=$1
FICHERO=$2
# Contamos las líneas de FICHERO en las
# que aparece CADENA
grep $CADENA $FICHERO | wc -l
# Terminamos satisfactoriamente
exit 0
```

Parámetros de entrada

Los scripts se pueden invocar pasando parámetros de entrada. Para acceder a ellos se usan variables especiales dentro del script

- `$#` almacena el número de parámetros que se han pasado
- `$0` almacena el nombre del propio script
- `$1, $2, $3, ...` almacenan los parámetros, por orden
- `$*` almacena todos los parámetros como una sola cadena

`shift`

Descarta el primer parámetro y desplaza los demás una posición

```
VAR1=$1
VAR2=$2
```

Es lo mismo que

```
VAR1=$1
shift
VAR2=$1
```

`read`

Lee uno o más parámetros por la entrada estándar y los almacena en variables

```
read VAR1 VAR2 VAR3 ...
```

Ejemplo con funciones

```
#!/bin/bash

function sum {
  echo $(($1+$2))
  return 0
}

div () {
  if [ $2 = 0 ]
    then
    echo "ERROR. Denominador = 0"
    return 1
  fi
  echo $(($1/$2))
  return 0
}

echo "Dime dos números: "
read NUM1 NUM2

echo "Suma: "`sum $NUM1 $NUM2`

RES_DIV=`div $NUM1 $NUM2`
if [ $? = 1 ]
then
  echo $RES_DIV
  echo "No se puede dividir"
else
  echo "Division: "$RES_DIV
fi
```

Almacenamiento de datos

En UNIX la información puede estar almacenada en uno o más dispositivos. Estos dispositivos se montan de manera ordenada para formar un árbol de directorios único. En dicho árbol se pueden combinar dispositivos de distinto tipo (discos duros, DVDs, discos virtuales…) y de distintas organización interna (sistema de ficheros). Al final todos ellos se gestionan en modo bloque o en modo carácter.

Algunos dispositivos se pueden además dividir en particiones para gestionar mejor su espacio.

Los dispositivos se pueden combinar mediante mecanismos hardware o software para crear entidades más sofisticadas (RAID, volúmenes lógicos…).

Gestión de dispositivos

1) **Dar formato al soporte**
   Separación física entre sectores, pistas, etc. Casi nunca necesario.

2) **Particionado**
   División del disco en zonas asignables a diferentes sistemas de ficheros. No necesario para CDs y DVDs.

3) **Creación del sistema de ficheros**
   Creación de las estructuras lógicas de un formato específico de organización de datos. Realizado sobre particiones (discos duros) o sobre dispositivos enteros (CDs/DVDs).

4) **Montaje y uso del soporte**

En el directorio `/dev` se encuentran los ficheros especiales para manejar los diferentes dispositivos de almacenamiento.

Una partición es una subdivisión física de la superficie de un disco duro. Dentro de una partición se puede crear un sistema de ficheros. Normalmente para poder almacenar información ordenada (sistema de ficheros), un disco duro debe tener al menos una partición. El espacio no particionado solo podrá ser accedido “en crudo” (raw mode).

La información relativa a las particiones se guarda en una tabla (tabla de particiones) al comienzo del disco (hay un espacio reservado para ello). Indica dónde comienza y termina cada partición e incluye información adicional como el sistema de ficheros o el punto de montaje.

Sistemas de ficheros

El sistema de ficheros organiza la información almacenada en un dispositivo en ficheros y directorios, almacenando permisos y otras propiedades.

- **Sistemas tradicionales**: almacenan datos y metadatos. Gestionan el espacio libre de la manera más eficiente posible. E.g.: ext2, ufs, hfs, FAT32.

- **Sistemas transaccionales**: extienden los anteriores añadiendo logs de operaciones y/o semántica transaccional. E.g.: ext3, ext4, ntfs, hfs+.

Los sistemas de ficheros se pueden montar manualmente con los mandatos `mount` y `unmount` o bien de forma automática gracias a determinados ficheros de configuración que contienen tablas de montaje.

Las cuotas de disco serán las que limitan el espacio en disco de cada usuario/grupo tanto en número máximo de archivos (i-nodos) como en número máx de bloques.

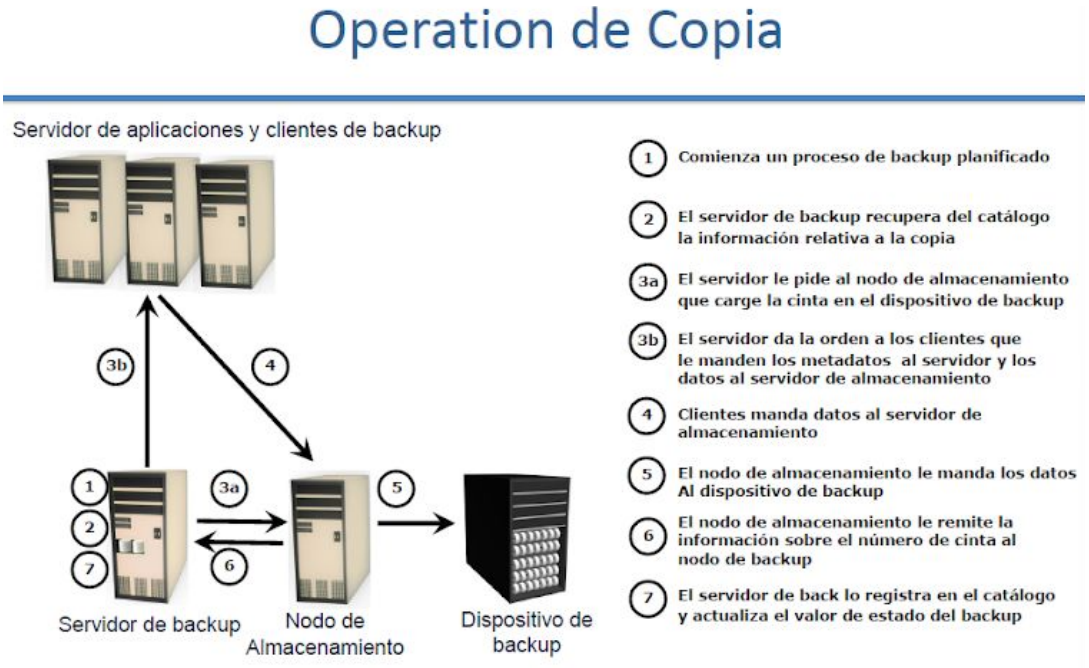
Backup

Copia de seguridad de determinados datos de un sistema. Se habla de backups completos, donde se copia toda la información; de backups diferenciales, donde partiendo de un backup completo se copia la parte modificada desde que se hizo; y de backups incrementales, donde sólo se copian los archivos modificados. Lo más común es combinar ambas estrategias realizando por ejemplo un backup incremental cada día y un backup completo cada semana.

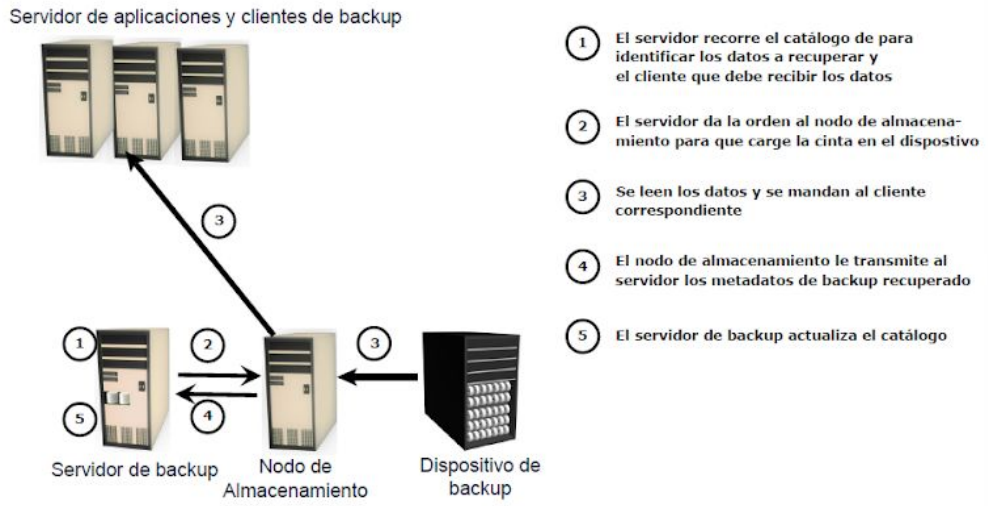
Los backups se pueden realizar en frío, deteniendo el sistema entre el inicio y final del backup, o en caliente, en el que se hace el backup con el sistema en producción. Este segundo se realiza típicamente a través de un fichero redo log que guarda la traza de acciones realizadas a partir del inicio del backup para luego aplicar las mismas operaciones una vez que la base de datos se ha copiado.

Tipos

- Backup de conexión directa.
- Backup vía LAN (Local AreaNetwork)
- Backup vía SAN (Storage AreaNetwork)
- Backup mixto
- Backup vía NAS (Network-AttachedStorage): Con/sin servidor



Operación de Recuperación



Recovery Point Objective (RPO): periodo máximo de tiempo en el cual se han podido ver afectados datos antes de un accidente.

Recovery Time Objective (RTO): periodo máximo de tiempo en el que es asumible tener los sistemas de información parados después de un accidente.

Plan de continuidad del negocio: indica la exposición de la organización a amenazas internas y externas y las contramedidas para prevención y recuperación.

Plan de recuperación ante desastres: determina los pasos a realizar para llevar a cabo las acciones de recuperación antes un incidente.

Una herramienta de backup sencillo es rsync, mandato que no es una solución de backup en sí misma pero que se puede utilizar para este caso ya que copia las diferencias entre los datos de origen y destino.

Ejemplo

Realizar un backup a un servidor remoto

- Directorio origen: `/home`
- Servidor destino: `server.org`
- Directorio destino: `/backups`

Mandato para realizar la copia:

```
rsync --recursive /home server.org:/backups/
```

¿Como hacer que se ejecute todos los días a las 4 de la madrugada?

Regla en `/etc/crontab`:

```
0 4 * * * rsync --recursive /home ...
```

Dispositivos redundantes

Los RAID son dispositivos virtuales compuestos por varios dispositivos físicos reales agrupados. Proporcionan redundancia y mejores prestaciones. Son invisibles para cada uno de los sistemas de ficheros que lo componen.

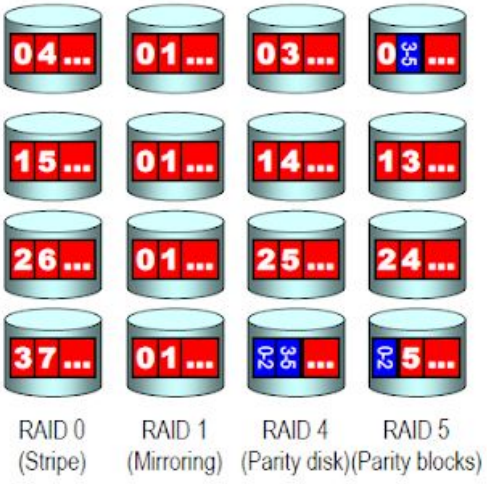
La tecnología RAID permite crearlo a través de hardware o software. En el primer caso es la controladora del dispositivo que proporciona el acceso al otros dispositivos. En el segundo el SO puede acceder de forma separada a los diferentes componentes del RAID (menos eficiente pero menor coste). Para este segundo caso existe el mandato mdadm que permite crear RAIDs, reconstruir RAIDs a partir de sus discos, monitorizar RAIDs creados y gestionar errores en RAIDs.

Redundant Array of Independent Disks

Tipos:

- Lineal: Concatenación
- RAID 0: Modo alternado
- RAID 1: Discos espejos
- RAID 4: Disco de paridad
- RAID 5: Bloques de paridad

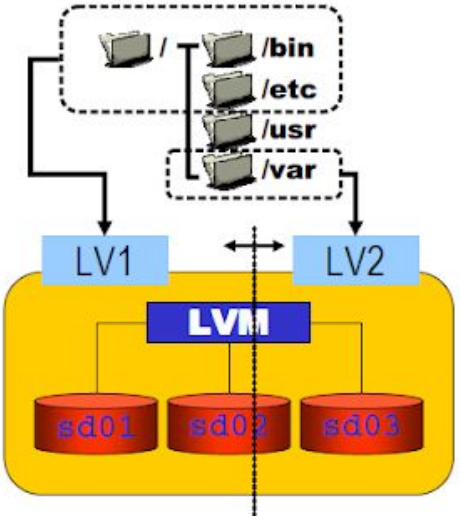
Puede haber discos de sobra (*spare disks*), que sirven de reserva



Para redimensionar un RAID es necesario poder redimensionar las particiones físicas o en caso contrario poder reemplazar los dispositivos físicos. En caso afirmativo los pasos a seguir son:

1) **Redimensionar cada partición que forma parte del RAID**
    a) Desconectar la partición
    b) Redimensionar la partición o reemplazar el dispositivo
    c) Añadir la partición de nuevo y esperar a que la sincronización termine
2) **Redimensionar el dispositivo virtual**
3) **Finalizada la sincronización, redimensionar el sistema de ficheros**

Los volúmenes lógicos son mecanismos para gestionar el espacio de forma dinámica. Presentan al usuario un espacio de direcciones contiguas, es decir, simulan un gran espacio de almacenamiento contiguo utilizando regiones de discos distintos. Esto permite redimensionar particiones y utilizar varios dispositivos como soporte.



LVM Tools es el gestor de volúmenes lógicos más común en GNU/Linux. Sus cuatro elementos básicos son: volúmenes físicos (discos o particiones), extensiones físicas, grupos de volúmenes y volúmenes lógicos.

Los pasos para crear un volumen lógico con LVM son:

1) Inicializar los volúmenes físicos que se vayan a usar
2) Crear un grupo de volúmenes
3) Crear uno o más volúmenes lógicos dentro del grupo
4) Crear el sistema de ficheros dentro de cada volumen lógico

Al finalizar, cada volúmen lógico se puede montar como una partición. Posteriormente se podrán agregar nuevos volúmenes físicos al grupo, cambiar el tamaño de los volúmenes lógicos, etc.

Tenemos tres particiones de 50GB (`sdb1`, `sdc1` y `sdd1`) y queremos crear dos volúmnes lógicos de 60GB y uno de 10GB

Inicializamos los volúmnes físicos

```
pvcreate /dev/sdb1 /dev/sdc1 /dev/sdd1
```

Creamos un grupo (llamado `dataserver`)

```
vgcreate dataserver /dev/sdb1 /dev/sdc1 /dev/sdd1
```

Creamos los tres volúmenes lógicos

```
lvcreate --name data1 --size 60G dataserver
lvcreate --name data2 --size 60G dataserver
lvcreate --name data3 --size 10G dataserver
```

Ahora podemos acceder a los volúmnes lógicos como si fuesen particiones de disco. Sus ficheros especiales son `/dev/dataserver/data1`, `/dev/dataserver/data2`, etc.

Ampliar su tamaño

```
lvextend -L15G /dev/dataserver/data3
```

Reducir su tamaño

```
lvreduce -L40G /dev/dataserver/data1
```

Crear un sistema de ficheros

```
mkfs.ext4 /dev/dataserver/data2
```

Montar el volúmen en el árbol de directorios

```
mount /dev/dataserver/data2 /mnt
```

¿Tiene sentido construir uno o mas volúmenes lógicos sobre un dispositivo RAID?
Depende del contexto

- Pros: Ventajas de tolerancia a fallos del RAID
- Contras: Posible pérdida de rendimiento (sobre todo si RAID software)

¿Tiene sentido construir un RAID a partir de volúmenes lógicos?
NO
¿Por qué?
La abstracción del espacio de disco en volúmenes lógicos anula las ventajas de tolerancia a fallos del RAID. Además se limita enormemente la flexibilidad de los volúmenes lógicos

Arranque del sistema y servicios básicos

Arranque del sistema

1) Arranque del sistema y comienzo de ejecución del firmware.
2) Inicio del cargador de arranque (bootloader).
3) Carga e inicio del núcleo del sistema operativo.
4) Arranque del proceso `init`.
5) Puesta en marcha de servicios, sistemas de ficheros, terminales, etc.

Bootloader

Normalmente almacenado en disco, es lanzado por el firmware del sistema y carga el núcleo en memoria y le transfiere el control.

En las distribuciones de Linux sobre arquitecturas Intel el cargador de arranque por defecto es GRand Unified Bootloader (GRUB). El proceso de carga se divide en dos etapas: una en el MBR o VBR y la otra en /boot/grub.

Initial RAM disk (initrd)

Se trata de un disco RAM temporal con utilidades y módulos para montar el sistema de ficheros raíz en los sistemas operativos Linux. Contiene un subsistema Linux mínimo: mandatos básicos, módulos del kernel, init, configuración de hardware mediante udev, etc.

Inicialización del kernel

1) Detectar la memoria disponible y reservarse algo.
2) En Linux, descomprimir initrd y montarlo.
3) Detectar y configurar el hardware.
4) Montar el sistemas de ficheros raíz (normalmente sólo lectura).
5) Lanzar procesos espontáneos internos (kjournald, kswapd, etc.).
6) Lanzar el proceso init.

Proceso init

Es el padre de todos los procesos. Tiene PID 1 y es el encargado de arrancar todos los servicios y dar acceso al sistema. Por defecto situado en /sbin/init.

El código de la secuencia de arranque no es parte del proceso en init en sí mismo, sino que se implementa como una colección de scripts que son ejecutados por init de acuerdo con unas reglas preestablecidas.

Cada uno de estos scripts es configurado por los ficheros de reglas situados en /etc/inittab en los cuales también se puede hacer referencia a scripts de inicialización rcn (donde n indica el runlevel).

Una vez termina la secuencia de arranque, el proceso init no termina (nunca lo hace), sino que queda a la espera de la terminación de cualquier proceso hijo para capturar su valor de retorno y evitar que se quede zombie; y se queda a la espera de alguna señal que indique que debe reconfigurarse el sistema.

El proceso init no aprovecha la posibilidad de arranque concurrente de determinados servicios. En sistemas más recientes se organizan los servicios en trabajos y/o tareas especificando dependencias entre éstos para habilitar una secuencia de arranque no secuencial.

Secuencia de arranque multiusuario

1) Comprobar raíz y montar en modo lectura-escritura.
2) Comprobar otros sistemas (/etc/fstab) y montarlos.
3) Activar el área de intercambio (swap).
4) Limpieza (e.g: /tmp).
5) Arrancar demonios locales.
6) Configurar la red y arrancar demonios de red.
7) Habilitar el login.

Modo monousuario

También llamado modo de mantenimiento, modo de arranque especial para la solución de problemas. Consola root, sin red ni login.

Secuencia de apagado del sistema

1) Notificación a los usuarios del apagado con un mensaje.
2) Envío de la señal SIG_TERM a todos los procesos en ejecución.
3) Proceso init comienza la secuencia de terminación de servicios.
4) Paso a modo monousuario:
    a) Se envía SIG_KILL a los procesos que al cabo de 5 segundos del envío de SIG_TEM aún no terminaron.
    b) Se desconecta a los usuarios y se bloquea el acceso.
5) Sincronización de los sistemas de fichero y desmontado.
6) Apagado o reinicio.

Systemd

Es un conjunto de demonios de administración de sistema, bibliotecas y herramientas diseñados como una plataforma de administración y configuración central para interactuar con el núcleo del Sistema operativo GNU/Linux. Se trata de un programa init.

systemctl: es la herramienta que nos permite interactuar con systemd. Permite hacer operaciones sobre servicios, consultar el estado del sistema y consultar y editar el contenidos de las units (services) y targets.

journalctl: permite examinar información de log de aplicaciones y kernel.

Udev

Sistema para gestionar nodos de dispositivo en /dev. Está compuesto por la biblioteca libudev, el demonio udevd (reglas en /etc/udev/rules.d), la herramienta de diagnóstico udevadm que permite conocer los atributos asociados a un dispositivo y un socket de comunicación llamado netlink.

El demonio crea los dispositivos bajo demanda (cuando los “descubre”), carga el módulo manejador correspondiente, crea el fichero especial, …

D-bus

Bus de sistema que proporciona mecanismos IPC (inter-process communication). Permite intercambio de mensajes entre aplicaciones basado en un esquema de emisor/suscriptor.

Existe un demonio de sistema (dbus-daemon) y otro por sesión de usuario para dar servicio al resto de aplicaciones. La funcionalidad de bajo nivel se implemente en la biblioteca libdbus.

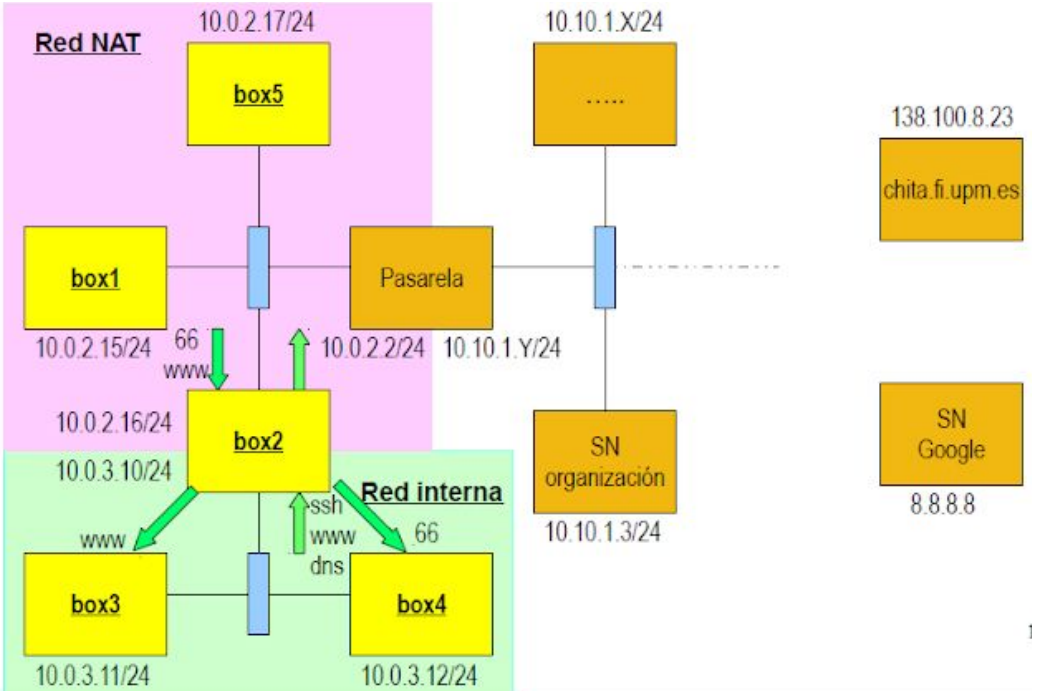
D-bus se puede utilizar sin el demonio, pero entonces sólo podemos establecer conexiones punto a punto (y no es necesario especificar el nombre del bus al que conectarnos).

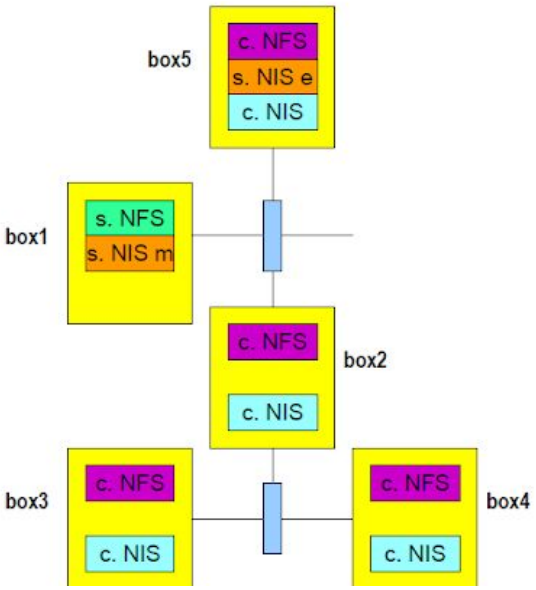
Administración de la red

Introducción

Actualmente es inconcebible pensar en una máquina “desconectada”, sin embargo la administración de una red es una labor muy compleja. Incluye diseño y despliegue de la arquitectura hardware, diseño y administración de la arquitectura software, despliegue de servicios de red, supervisión de la seguridad…

Para estudiar un caso práctico se plantea una red de dos subredes: una con total conectividad entrante y saliente y otra con conectividad entrante y saliente restringida a ciertos puertos. Las direcciones IPs en red privada no son conocidas fuera de la misma. Además se desplegará un servicio SSH, NFS y NIS.





Configuración de una máquina

1) **Configuración IP estática temporal**
   Se configura a través de los mandatos `ip address add`, `ip link set` y `ip route add`.
   a) Configura IP, máscara de red y dirección de broadcast.
   b) Activa la interfaz
   c) Añade la ruta correspondiente a la pasarela

2) **Configuración IP estática persistente**
   Se configura en `/etc/network/interfaces` gracias los mandatos `auto` (habilita interfaz en el boot) y `iface` (define interfaz).

3) **Configuración DNS**
   Se configura en `/etc/nsswitch.conf` el mecanismo de traducción para cada tipo de recurso y en `/etc/resolv.conf` se especifican servidores DNS y dominios para nombres incompletos.

4) **Configuración IP dinámica basada en DHCP**
   Se obtiene una configuración temporal mediante el mandato `dhclient` y se configura permanentemente en el fichero `/etc/network/interfaces` a través gracias a iface.

Configuración de una subred

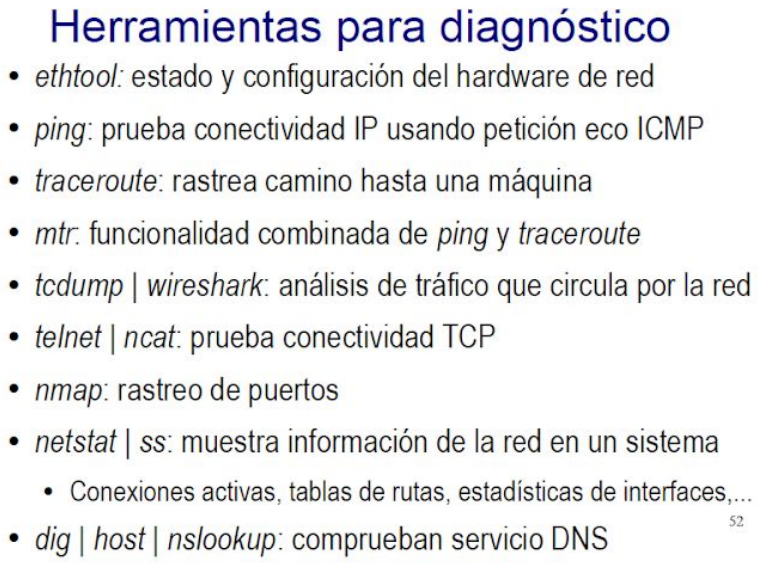
5) **Configuración de la conectividad entre las 2 redes**
   a) Configurar nuevas interfaces y actualizar tabla de rutas (`/etc/network/interfaces`)
   b) Activar en box2 la retransmisión de paquete entre subredes

6) **Filtro de tráfico en el equipo retransmisor**
   Se realiza con el módulo netfilter y se configura a través de iptables. En ellas se declara las reglas que se aplican al tráfico de paquetes. Se añade en /etc/network/interfaces un mando para cargar la iptable.

7) **IP masquerading**
   El tráfico de la subred aparece como emitido por un único nodo (el que efectúa de pasarela con las otras redes). Ello asegura que las direcciones IP internas no salgan de la subred y permite usar direcciones IP privadas dentro de la misma, lo cual facilita el diseño de redes más seguras. Se consigue a través de la iptable nat cadena POSTROUTING.

   También existen los TCP wrappers que permiten filtrar el acceso a servidores de determinados servicios con sus listas /etc/hosts.allow y /etc/hosts.deny.

8) **Destination NAT**
   Permite el acceso desde redes externas a determinados servicios internos. Se accede a éstos a través del IP del nodo intermediario.



Despliegue de servicios de red

9) **SSH**
   El administrador puede necesitar hacer login en un equipo remoto para ejecutar un mandato en él o copiar un fichero a/desde él. SSH proporciona autenticación y comunicación cifrada.
   También se puede configurar un túnel SSH: port forwarding. Permite que si hay un servidor web en un Equipo 1 en la Red 1 y tengo una cuenta en un Equipo 2 en la Red 1 pero me encuentro actualmente en una Red 2 pueda conectarme al servidor web del Equipo 1 a través de mi login del Equipo 2.

10) **NFS**

Permite que equipos puedan compartir ficheros (e.g.: /home). Sistema de ficheros distribuido estándar para sistemas UNIX. El servidor exporta un subconjunto de sus ficheros y el cliente realiza un montaje remoto de los mismos y añade a posteriori una entrada del punto de montaje asignado en /etc/fstab.

Los ficheros exportados se encuentran en /etc/exports y el mandato exportfs -ra hace efectos los cambios realizados.

11) **NIS**

Gestión global de recursos (e.g.: cuentas de usuario). Es un protocolo de envío de datos de configuración tales como nombres de usuarios y hosts entre computadoras sobre una red.

En el servidor se creará el nombre de dominio en /etc/defaultdomain, se configura en /etc/default/nis → NISSERVER=master, se restringe el acceso en /etc/ypserv.securenets y se modifica el /var/yp/Makefile si queremos que las contraseñas estén también en el repo. Finalmente se vuelca la información con el mandato yinit. Habrá que reiniciar el servidor rpcbind y NIS para que los cambios surjan efecto.

En el cliente se configurará dónde se encuentran el servidor y el esclavo en /etc/yp.conf y se especificará /etc/nsswitch.conf qué información está en el servidor NIS.

Se configura un esclavo (primero como cliente y luego como maestro) para que en caso de que el servidor se caiga responda el esclavo por él.

Gestión de kernel y dispositivos

Si se necesita personalizar el kernel del sistema operativo hay cuatro método para lograrlo:

1) **Pasar opciones al cargador de arranque**
   [Ya visto]

2) **Modificar parámetros del kernel**
   Se encuentran en /proc/sys y son accesibles a través del comando sysctl. Se pueden modificar de forma no permanente o de forma permanente (/etc/sysctl.conf).

3) **Compilar una nueva versión del kernel que incluya las nuevas funcionalidades**
   Se debe configurar opciones, compilar kernel y módulos, copiar el kernel y el System.map a /boot, configurar el GRUB y crear initrd, copiar módulos a /lib/module/$(uname -r) y calcular dependencias entre módulos.

4) **Incorporar módulos al kernel**
   Linux es un SO con un kernel monolítico pero permite añadir funcionalidad en tiempo de ejecución mediante módulos. Éstos son ejecutados dentro del kernel como si fueran una DLL de éste.

Gestión de software

El modelo de distribución de software basado en repositorios es un punto fuerte de Linux y facilita el desarrollo de software en comunidad. La unidad de distribución es el "paquete", el cual incluye script para instalar, configurar, activar, …

Los gestores de paquetes son los encargados de aportar funcionalidad en relación con los paquetes. Listar paquetes, buscar paquetes, obtener información de ellos, actualizarlos, instalarlos, eliminarlos… Existen dos tipos de gestores de paquetes:

- **Bajo nivel** (dpkg): no interaccionan con el repositorio, no resuelven dependencias entre paquetes.

- **Alto nivel** (apt-get): trabajan con repositorios, sí dependencias entre paquetes. Se complementa con apt-cache para preguntar sobre metainformación de paquetes y apt-file para preguntar sobre ficheros incluidos en paquetes.

WINDOWS

Active Directory

Grupo de trabajo: un ordenador o más de uno conectado a una misma red LAN pero que no están unido por un dominio. Cada uno de ellos es independiente al resto.

Dominio: colección de objetos que comparten la misma base de datos. Esto permite administrar equipos de un grupo de trabajo sin tener que acceder individualmente a cada uno de ellos.

Servicios de dominio Active Directory (AD DS): servicio integrado en los servidores Windows pero no instalado por defecto, habilita un servidor (llamado controlador de dominio) para ser la base de datos de identidades de un dominio y proporciona todas las herramientas para interactuar con ésta. Fuente centralizada de autenticación y autorización en el dominio, y por tanto necesitará replicación.

Sitio: representa la estructura física o topología de una red. Por definición, un sitio es una colección de subredes conocidas.

Replicación: en Active Directory cada controlador de dominio posee una copia del AD DS. Cuando uno de ellos realiza un cambio, este cambio debe ser propagado a las copias de los otros controladores. Se estima en 15 segundos para dominions en el mismo site y en 15 minutos para dominios en diferentes sites.

Objetos: todo lo que hay en un Active directory es un objeto. Usuarios, grupos, ordenadores, sites, subredes, etc.; todos estos son objetos con sus respectivas propiedades.

Schema: es una representación de una clase de objetos del sistema. Indican por ejemplo cuáles son las propiedades de cierto objeto.

Unidad organizacional: sirve para agrupar objetos similares en el Active Directory, sobre todo usado para usuarios y computadores. Sirven para enlazar objetos con Group Policies y delegar el control sobre ellos.

Group Policy: se utilizan para configurar ajustes de usuarios y equipos. Permiten configurar a la vez múltiples equipos que se encuentren en una misma unidad organizacional. Por defecto todo dominio tiene un Default Domain Policy y un Default Domain Controllers Policy.

Default Domain Policy: se crea a la vez que el dominio es creado. Contienen ajustes sobre usuarios y equipos que se aplicarán a todos los miembros del dominio.

Default Domain Controllers Policy: son policies que sólo afectan a los controladores de dominios. Cuando se crea un nuevo controlador de dominio automáticamente se le aplica la configuración especificada en este documento.

Bosque: es una instancia única de Active Directory. Dentro de un bosque se puede encontrar uno o múltiples dominios que comparten el mismo esquema.

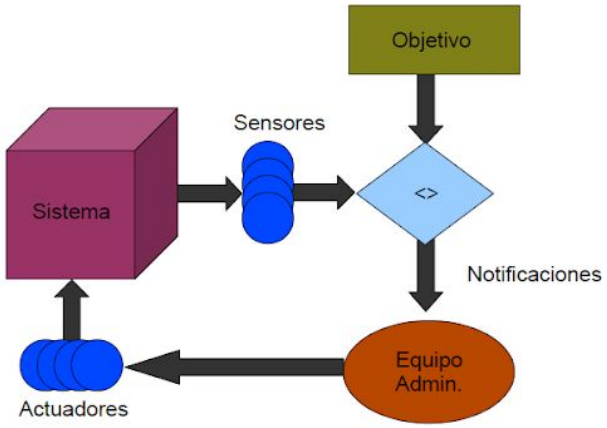
Catálogo global: contienen información sobre cada objeto de cada dominio en un bosque con varios dominios. El catálogo se almacena en los controladores de dominios que han sido habilitados como servidores de catálogo global y se replica a través del Active Directory. Así, un catálogo en un dominio contendrá toda la información acerca de los objetos de dicho dominio pero sólo cierta información de los objetos de otros dominios. Es aconsejable tener uno en cada site.

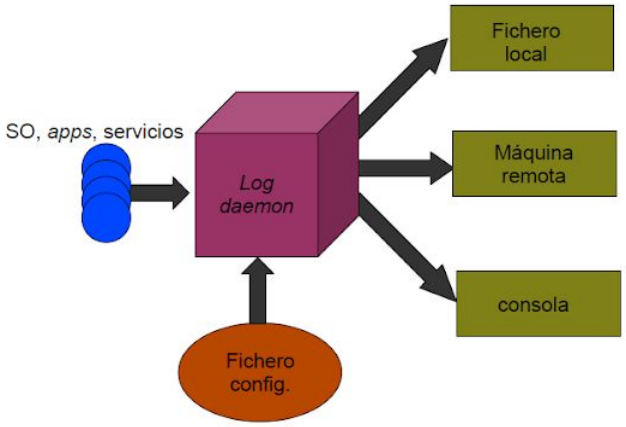
Confianza: conexión entre dominios para acceder a sus recursos (como servidores o aplicaciones). Todos los dominios dentro de un mismo bosque poseen vínculos de confianza entre ellos.

Árbol: si construyes uno o más dominios dentro de un mismo bosque que tienen un espacio de nombres contiguos o/y comparten el mismo esquema, creas un árbol. Un espacio de nombres contiguo es un dominio que comparte el mismo nombre de dominio raíz. E.g.: enzoferey.com y api.enzoferey.com

MONITORIZACIÓN

Un sistema de monitorización se encarga de dar los medios a un administrador para que supervise el correcto funcionamiento del servicio y sobre todo lo notifica cuando ocurre un problema. El problema es que hay muchos aspectos que pueden fallar, desde hardware hasta software, hasta sobrecarga, o incluso una falta de corriente.





Monitorización del procesador

Parámetros relevantes: uso de los procesadores, longitud de la cola de listos, número de cambios de contexto, números de interrupciones, número de procesos creados.

uptime, vmstat, mpstat, procinfo, ps, top, iostat, pidstat

Profiling: estadística de gasto de recursos de módulo software. Útil para la detección de partes a optimizar.

Monitorización de la memoria

Parámetros relevantes: tamaño de memoria física y su estado, tamaño disponible en paginación (swap), estadística sobre operaciones (fallos de página, páginas leídas/escritas…).

free, swapon -s, sar, pmap

Monitorización de los discos

Parámetros relevantes: bloques leídos/escritos (total y cuántos/segundo), porcentaje de ocupación del disco, número de operaciones que pueden juntarse, tamaño medio de cola de espera del disco, tiempo medio de servicio a una petición.

vmstat, iostat -x, sar -b, sar-d, pidstat -d

Monitorización de la red

Parámetros relevantes: paquetes enviados/recibidos, bytes enviados/recibidos, estadísticas sobre errores como paquetes erróneos recibidos, errores de transmisión, falta de buffers, colisiones, ….

netstat, sar -n DEV, sar -n EDEV, ss

Monitores integrados

Se trata de una colección de scripts independientes que hacen una tarea específica de monitorización, en general dentro del mismo software. No proporcionan una solución adecuada ya que no se posee una visión general, suelen significar que falta una estrategia global de notificación, no es tolerante a fallos y se suelen hacer comprobaciones redundantes. Además dificulta la generación de informes ya que la información se encuentra distribuida.

Monitorización de sistemas distribuidos

Se requiere la supervisión de recursos y servicios de múltiples nodos. Algunos de los parámetros relevantes serán la disponibilidad del nodo (ping), la actividad de un servicio, el estado de los recursos locales del nodo (accesible a través del mandato free a través de ssh), agentes “sensores” instalados en el nodo, modo de muestreo activo vs pasivo y el posible uso de SNMP. Habrá que tener en cuenta las interdependencias entre nodos para no gastar recursos en un nodo que no nos dará ninguna información sobre el problema.

Monitorización caja negra vs blanca

Caja negra: comprueba disponibilidad de recursos o servicios y en cuanto detecta que alguno de ellos no está disponible salta una alerta. No anticipa fallos inminentes ni evolución del fallo.

Caja blanca: obtiene medidas de los sensores en cada uno de los nodos y gestiona y almacena esos datos para que puedan ser posteriormente visualizadas y analizadas. De esta forma se pueden establecer avisas según la evolución de ciertas medidas y así anticipar fallos.

Monitorización distribuida

En sistemas de gran escala, por ejemplo un cluster, requiere múltiples nodos de monitorización. Se establece así una jerarquía de monitores que agregan y filtran la información.